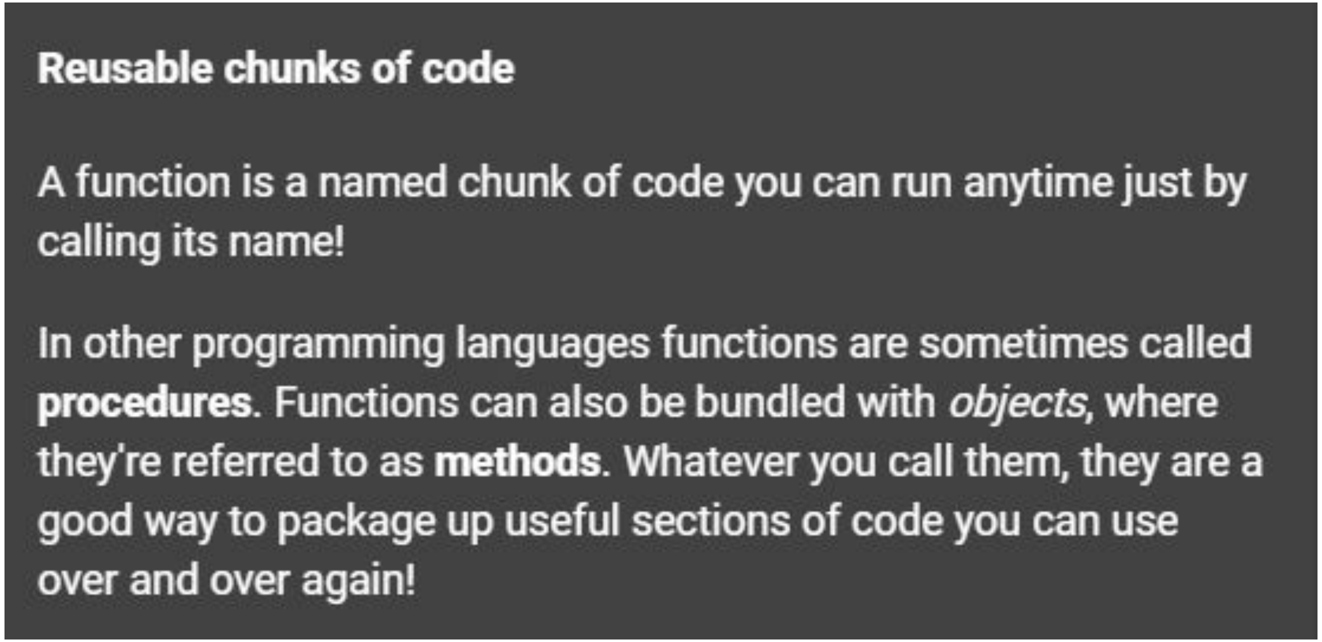
Defining and Using Functions

An extension for Mission 4

What is a function?

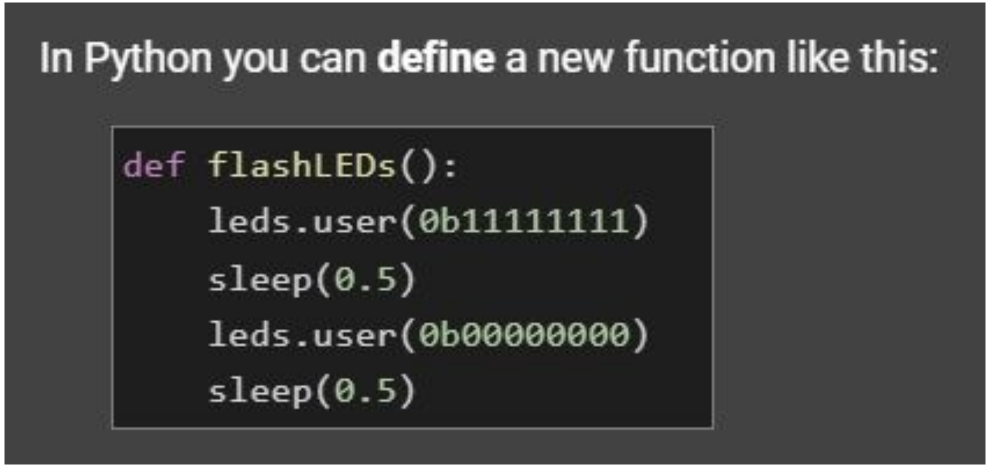


Reusable chunks of code

A function is a named chunk of code you can run anytime just by calling its name!

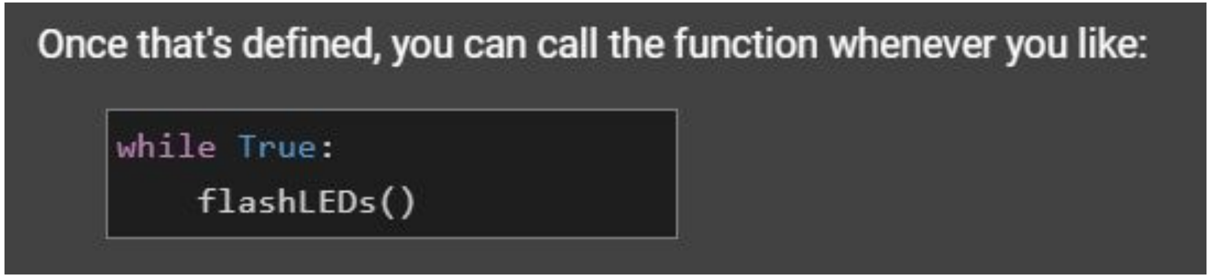
In other programming languages functions are sometimes called **procedures**. Functions can also be bundled with *objects*, where they're referred to as **methods**. Whatever you call them, they are a good way to package up useful sections of code you can use over and over again!

What is a function?



In Python you can **define** a new function like this:

```python
def flashLEDs():
    leds.user(0b11111111)
    sleep(0.5)
    leds.user(0b00000000)
    sleep(0.5)
```



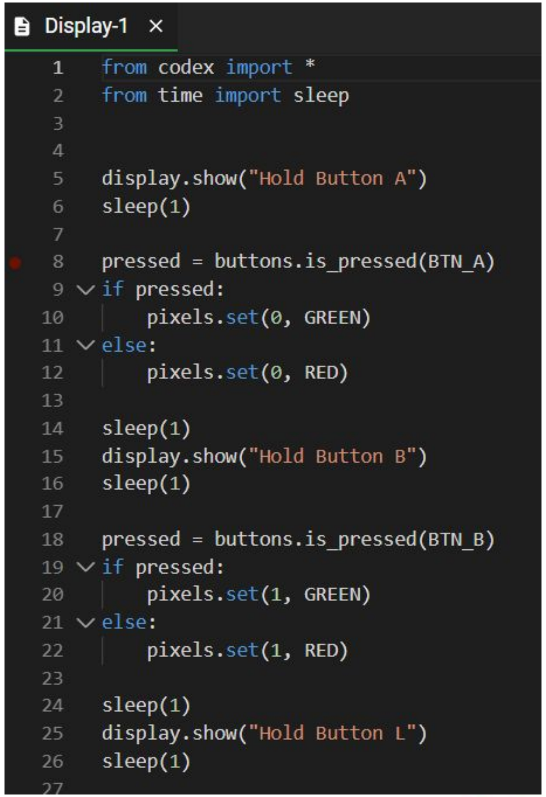
Once that's defined, you can call the function whenever you like:

```python
while True:
    flashLEDs()
```

Try it out in your Mission 4 code!

Open your Display code (if not already open).

It may look like similar to this, but it could be different.



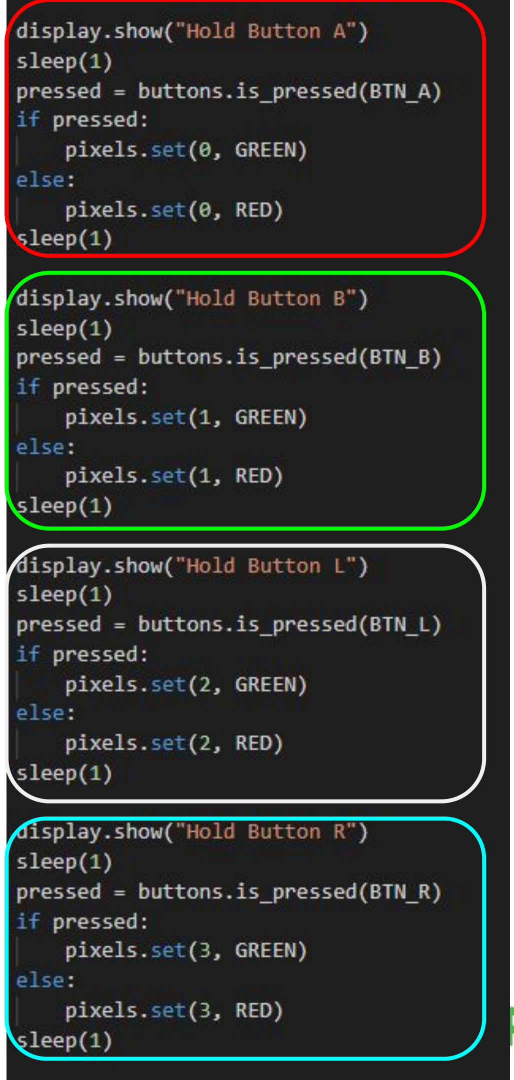
Display-1

```
from codex import *
from time import sleep


display.show("Hold Button A")
sleep(1)

pressed = buttons.is_pressed(BTN_A)
if pressed:
    pixels.set(0, GREEN)
else:
    pixels.set(0, RED)

sleep(1)
display.show("Hold Button B")
sleep(1)

pressed = buttons.is_pressed(BTN_B)
if pressed:
    pixels.set(1, GREEN)
else:
    pixels.set(1, RED)

sleep(1)
display.show("Hold Button L")
sleep(1)
```

Identify sections of code

Look through your code and find sections that could be functions.

- You probably have four sections in your code.
- Each section is similar but asks for a different button push and a lights a different pixel

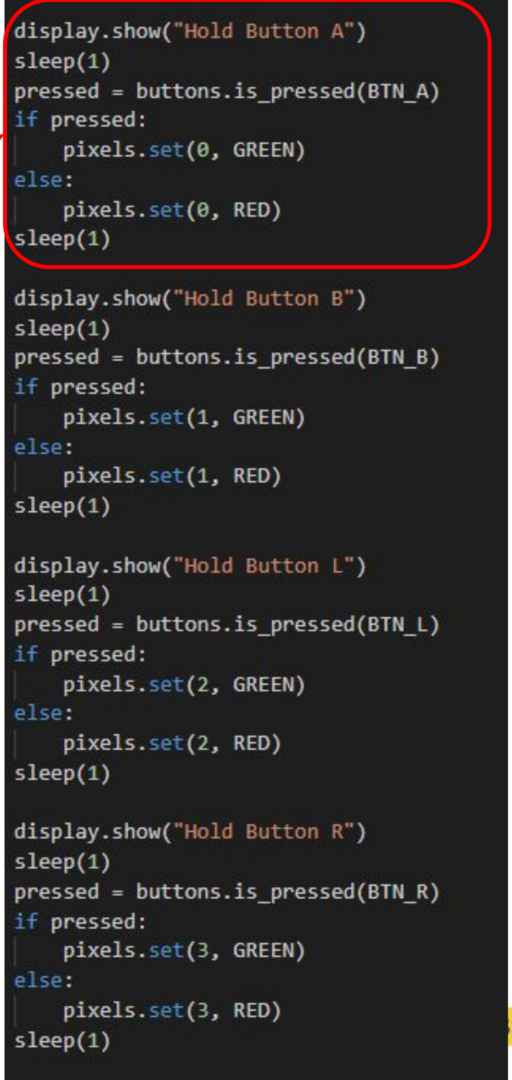


```
display.show("Hold Button A")
sleep(1)
pressed = buttons.is_pressed(BTN_A)
if pressed:
    pixels.set(0, GREEN)
else:
    pixels.set(0, RED)
sleep(1)

display.show("Hold Button B")
sleep(1)
pressed = buttons.is_pressed(BTN_B)
if pressed:
    pixels.set(1, GREEN)
else:
    pixels.set(1, RED)
sleep(1)

display.show("Hold Button L")
sleep(1)
pressed = buttons.is_pressed(BTN_L)
if pressed:
    pixels.set(2, GREEN)
else:
    pixels.set(2, RED)
sleep(1)

display.show("Hold Button R")
sleep(1)
pressed = buttons.is_pressed(BTN_R)
if pressed:
    pixels.set(3, GREEN)
else:
    pixels.set(3, RED)
sleep(1)
```

Define a function

Create a function for the first section of code

- Functions typically are coded near the top of the program, under imports and variables
- Use a descriptive name for the function
- A function definition ends with a colon (:) – you are creating a block of code
- Don't forget to indent! – the shortcut for this is to highlight the text and press TAB

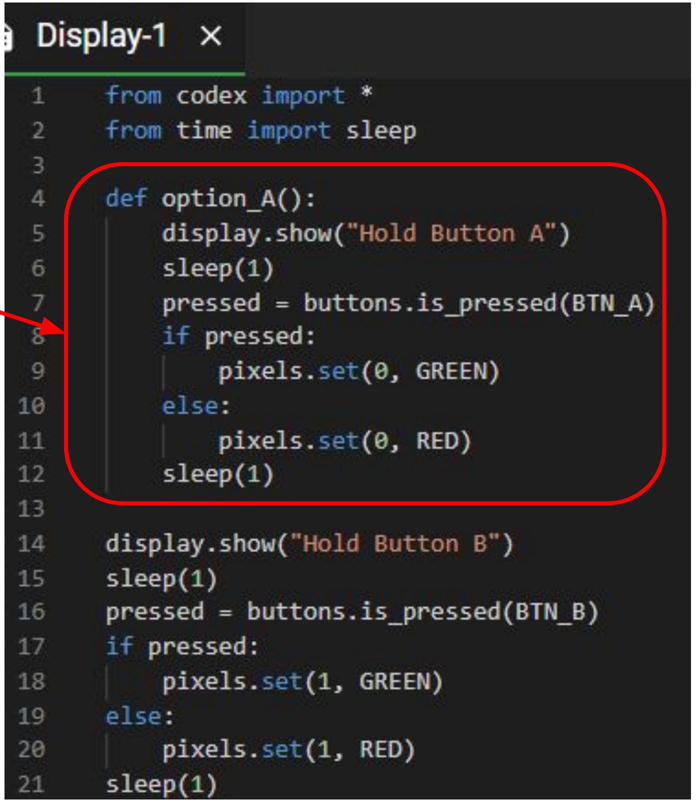


```
display.show("Hold Button A")
sleep(1)
pressed = buttons.is_pressed(BTN_A)
if pressed:
    pixels.set(0, GREEN)
else:
    pixels.set(0, RED)
sleep(1)

display.show("Hold Button B")
sleep(1)
pressed = buttons.is_pressed(BTN_B)
if pressed:
    pixels.set(1, GREEN)
else:
    pixels.set(1, RED)
sleep(1)

display.show("Hold Button L")
sleep(1)
pressed = buttons.is_pressed(BTN_L)
if pressed:
    pixels.set(2, GREEN)
else:
    pixels.set(2, RED)
sleep(1)

display.show("Hold Button R")
sleep(1)
pressed = buttons.is_pressed(BTN_R)
if pressed:
    pixels.set(3, GREEN)
else:
    pixels.set(3, RED)
sleep(1)
```

Define a function

Your function may look like this.

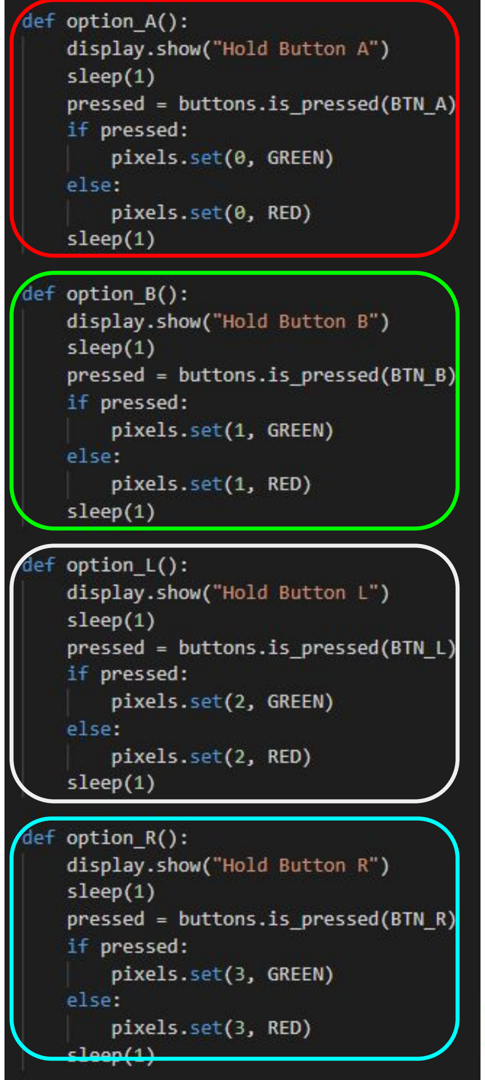
- Create functions for the other three sections of code



```
from codex import *
from time import sleep

def option_A():
    display.show("Hold Button A")
    sleep(1)
    pressed = buttons.is_pressed(BTN_A)
    if pressed:
        pixels.set(0, GREEN)
    else:
        pixels.set(0, RED)
    sleep(1)

display.show("Hold Button B")
sleep(1)
pressed = buttons.is_pressed(BTN_B)
if pressed:
    pixels.set(1, GREEN)
else:
    pixels.set(1, RED)
sleep(1)
```

Call a function

- Now you have functions for each task (or button press)
- Four functions for four tasks
- Is your indenting correct?
- Will your code work properly now? Why or why not?

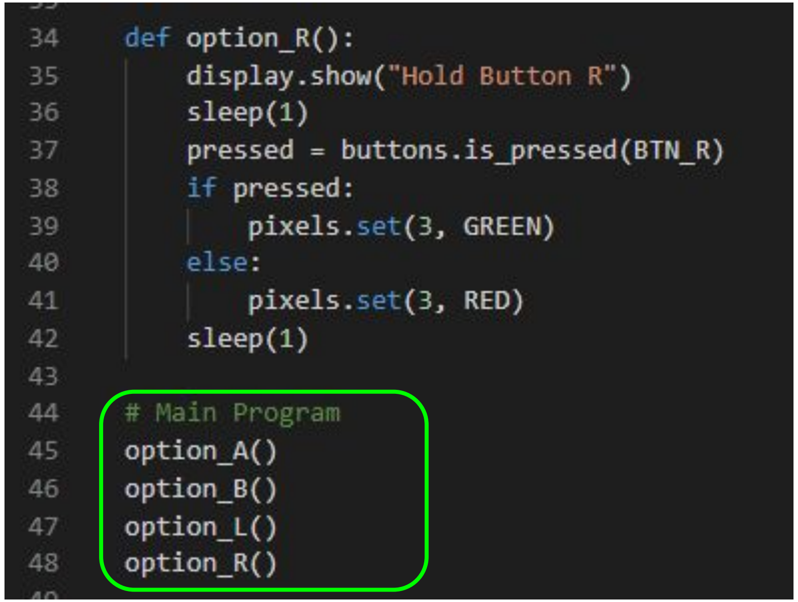


```
def option_A():
    display.show("Hold Button A")
    sleep(1)
    pressed = buttons.is_pressed(BTN_A)
    if pressed:
        pixels.set(0, GREEN)
    else:
        pixels.set(0, RED)
    sleep(1)

def option_B():
    display.show("Hold Button B")
    sleep(1)
    pressed = buttons.is_pressed(BTN_B)
    if pressed:
        pixels.set(1, GREEN)
    else:
        pixels.set(1, RED)
    sleep(1)

def option_L():
    display.show("Hold Button L")
    sleep(1)
    pressed = buttons.is_pressed(BTN_L)
    if pressed:
        pixels.set(2, GREEN)
    else:
        pixels.set(2, RED)
    sleep(1)

def option_R():
    display.show("Hold Button R")
    sleep(1)
    pressed = buttons.is_pressed(BTN_R)
    if pressed:
        pixels.set(3, GREEN)
    else:
        pixels.set(3, RED)
    sleep(1)
```

Call a function

- All of the code is in functions
- Functions have to be called for their instructions to run
- The great thing about functions is you can call them multiple times and in any order

Here is one example of calling functions

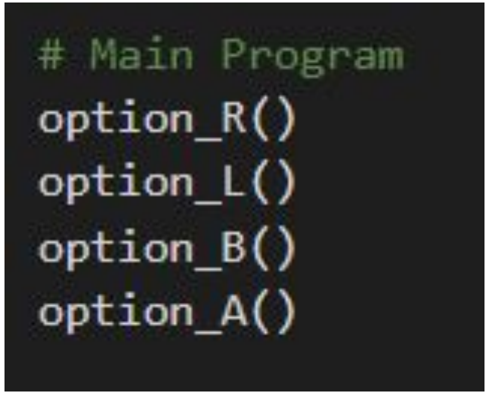


```
def option_R():
    display.show("Hold Button R")
    sleep(1)
    pressed = buttons.is_pressed(BTN_R)
    if pressed:
        pixels.set(3, GREEN)
    else:
        pixels.set(3, RED)
    sleep(1)

# Main Program
option_A()
option_B()
option_L()
option_R()
```

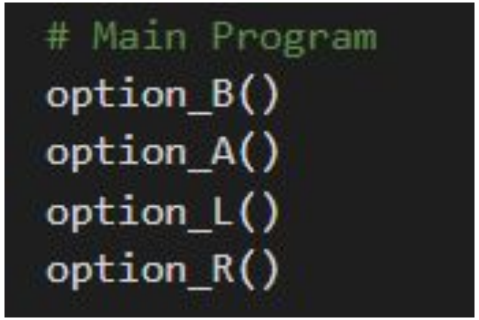
Call a function

Here are more examples. There are many possibilities!

- Function calls go BELOW function definitions
- A function call does NOT end with a colon (:)
- The functions will be run sequentially, in the order you call them



```
# Main Program
option_R()
option_L()
option_B()
option_A()
```

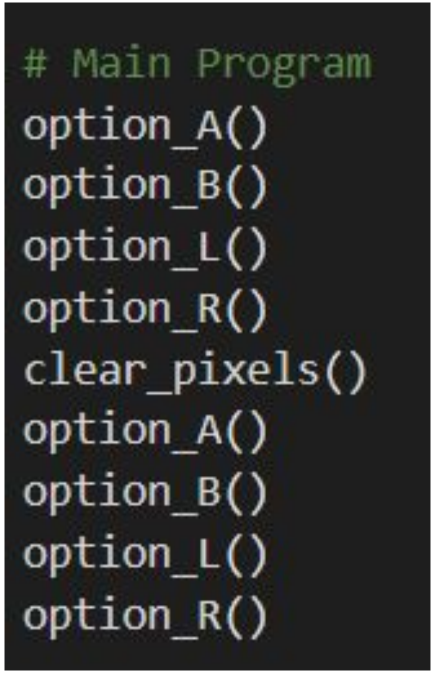


```
# Main Program
option_B()
option_A()
option_L()
option_R()
```

Mission 4 extension

Using functions opens up all kinds of possibilities. Try this:

- Create a function that turns off all pixels (set each to black)
- Call all four option functions
- Then call the function to turn off the pixels
- Then call the four option functions again, to replay the game

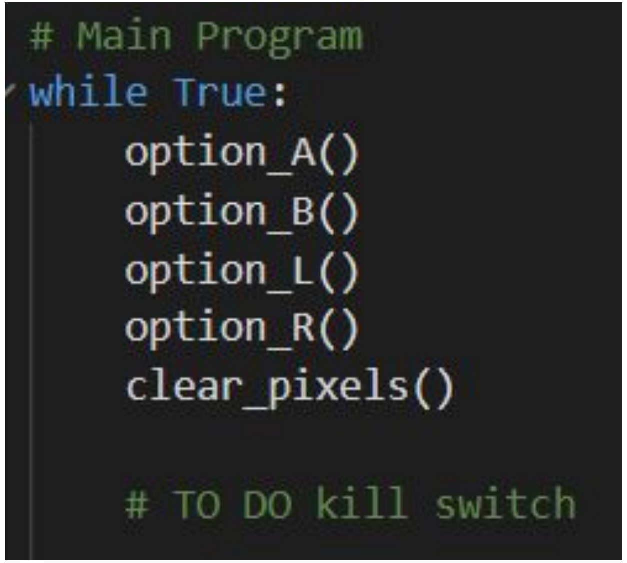


```
# Main Program
option_A()
option_B()
option_L()
option_R()
clear_pixels()
option_A()
option_B()
option_L()
option_R()
```

Mission 4 extension

More extensions:

- Put your code in a loop to play forever
- Add a “kill switch” to end the loop



```
# Main Program
while True:
    option_A()
    option_B()
    option_L()
    option_R()
    clear_pixels()

    # TO DO kill switch
```